Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 8, Issue, 03, pp. 28307-28313, March, 2016

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

ISSN: 0975-833X

RESEARCH ARTICLE

GOAL ORIENTED DEVOPS TRANSFORMATION FRAMEWORK – METRIC PHASED APPROACH

***Samer I. Mohamed**

Modern Science and Arts University, Faculty of Engineering, Electrical and Communication Department, Egypt

ARTICLE INFO

Article History:
Received 14th December, 2015
Received in revised form
20th January, 2016
Accepted 26th February, 2016
Published online 31st March, 2016

Key words:

Cloud computing,
Transformation framework,
Key Performance Indicator,
Critical Success Factor,
Maturity model.

ABSTRACT

With the introduction of the cloud computing and virtualization, a need for a new model that enables organizations to better satisfy their clients expectations from both infrastructure and services perspectives becomes a necessity (Bang *et al.,* 2013). DevOps is a collection of best practices and standards that formulate the required model that enables those organizations to evolve and adapt to the target spectrum. This paper provides a new innovative DevOps transformation framework adopting a quantitative or metric phased approach that can be utilized by any organization aims to transform delivery model into the DevOps model. The edge of the proposed framework is that it provides a structured and quantitative mechanism via different transformation phases to measure both the current state in the assessment phase via clear KPI (Key Performance Indicators) and CSF (Critical Success Factors), and then isolate the gaps covered within the assessment phase to move to the next state via clear transformation actions. Thus instead of following qualitative measures, like those adopted by other transformation frameworks (Le-Quoc, 2011), the proposed goal-oriented framework quantitatively measures any project/organization maturity using metric phased approach, against different capabilities, with different capability levels. This is done by reviewing the observed project/organization behaviors against the standard framework description described at each capability level.

Citation: Samer I. Mohamed, 2016. "Goal oriented DevOps transformation framework – Metric phased approach", International Journal of Current Research, 8, (03), 28307-28313.

INTRODUCTION

In the cloud world, the traditional approaches for application development and management prove not to accommodate with the increasing trends of clients demands and expectations. Thus a need for a new approach that enables IT organizations to ensure integrated software systems are always on and available to support clients requests while reducing expected risks, maximize the value and/or ROI (Return on investment) and minimize the TCO (Total Cost of Ownership). These challenges faced currently by many IT organizations pushed it towards adopting DevOps model. DevOps model is a collection of standards and best practices that help IT organizations to learn how to become a service provider to its business users, move quickly to meet changing business environments, enable a self-service mechanism for selecting and provisioning the IT services and use automation to deliver more with less costs. DevOps is an evolution in thinking with regards how IT services are delivered and supported. It is a continuation of some of the predecessor work in the areas of continuous integration and application life cycle management (ALM); therefore, it is rooted in the agile philosophy, which also attempts to bridge the traditional organizational process divide between development and operations teams (Akerele, 2013). The value behind DevOps lays in bridging the current gap between the different technical roles within the same team who work in silos. Thus, The DevOps approach is built around those who believe that the application of a combination of appropriate technology and attitude can revolutionize the world of software development and delivery especially these different roles share the same objective which is the delivery of a successful products under a stressful market conditions (Debois, 2011). The key for any organization to get benefit from the DevOps model is to follow the below best practices:

Automation: Faster release cycles, combined with massively scalable cloud environments, demand the ability to automate every aspect of the release process. Tools such as Puppet and Chef eliminate manual processes and replace them with simpler, standardized and highly repeatable software deployment methods.

Scripting and coding: Many system administrators are already comfortable using tools like Perl, a tool that was actually developed as a programming language for automating system administration.

****Corresponding author: Samer I. Mohamed,***
Modern Science and Arts University, Faculty of Engineering,
Electrical and Communication Department, Egypt.

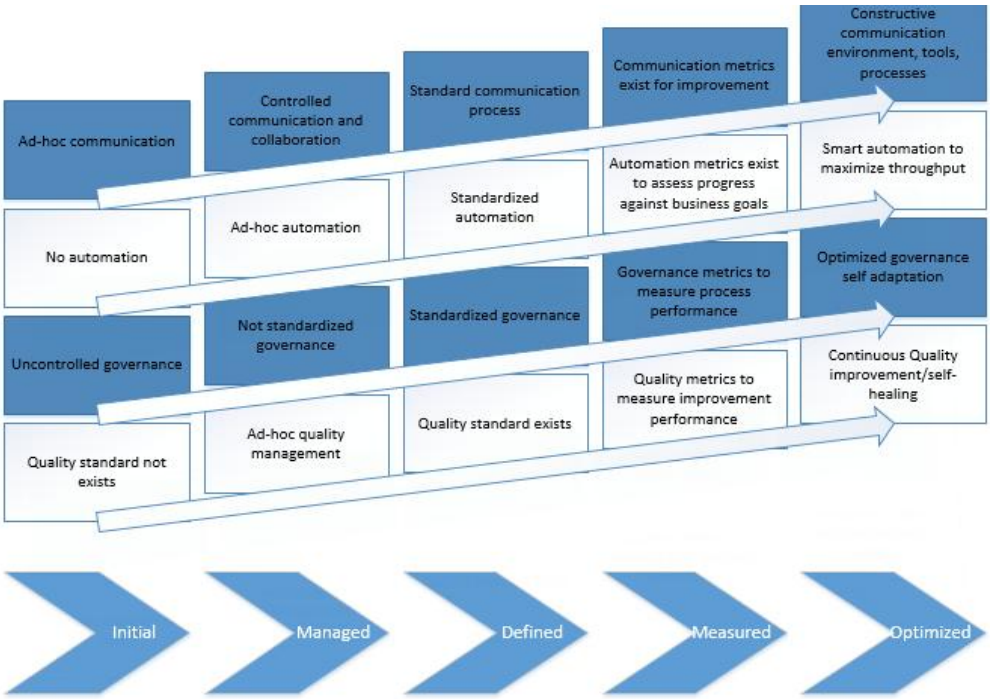

Fig. 1. DevOps maturity model

Infrastructure via APIs: Infrastructure APIs give Operations a standard framework for provisioning and configuring cloud-based infrastructure components. By offering access to these components, infrastructure APIs simplify the process of deploying and managing cloud-based applications allowing programmatic changes to the infrastructure.

Continuous learning: It's not enough for Development and Operations to agree to work together. They must also agree on the processes for experimentation and continuous improvement in delivering new applications and deploying updates to existing applications. DevOps works together to select and implement which platforms provide the best foundation for key activities such as automation and self-service delivery models.

Platform-based development: Platform as a service (PaaS) cloud computing services provide the entire infrastructure from hardware and operating systems to databases and middleware — required to develop and run applications. Developers can build applications faster and cheaper using a platform-based approach, but doing so requires the right tools and the right understanding of how to adapt their code to a PaaS environment.

Focus on organizational process alignment: According to [S. Stuckenberg], the most successful DevOps models focus on organizational issues and process alignment, including leadership. Especially these processes which impact the release delivery/cycle by smoothing the value gained from each process via clear KPIs and measurements.

Leverage internal talent: It's often easier and more efficient to identify members that already have the right skills to build a DevOps engine. Server and storage infrastructure support personnel, for example, are probably already well-versed in the scripting, configuration management and automation skills required to drive the "ops" side of DevOps.

Create a continuous feedback loop: In order to support continuous delivery and integration of applications, DevOps must also put the processes in place to collect and implement feedback – and not just from IT staff like business users and other involved stakeholders. The paper is organized as follows: section II gives a background for the DevOps maturity model where the transformation framework is based; section III provides the main construction of the proposed transformation framework; section IV Validate the new DevOps transformation framework via use case; section V is the conclusion for what has been presented in the paper (Schaefer *et al.,* 2013).

DevOps maturity model background

To enable organizations get the most outcomes/value from the DevOps, a transformation framework is proposed to first assess the current state of the organization/department, and then apply the proposed transformation framework according to the gap assessment results. This transformation framework will apply set of processes and best practices to make the organization operating model adaptable with the DevOps model. The proposed model is based on the maturity model introduced in (Samer, 2015). With five levels of maturity. Each level is assessed against 4 dimensions (Quality, Automation, communication/collaboration, and governance) as described in DevOps maturity model in Fig 1. To move from one any maturity level to the following one, organization needs to improve against the 4 dimensions. This proposed model is inherited from HP model [HP model] to cover the entire life cycle of the any delivered service E2E (End to End).

At the initial level of DevOps maturity, organizations are capable of doing ad hoc deployments of services when they are ready, but they have a hard time predicting when those services will be production quality and are dependent on the

actions of talented resources. At that level of maturity, there are many manual steps involved in deploying a new version of software service/application. As a result, a full service release cycle (inclusive of testing) can take days or even weeks to complete. The results of maintaining an initial level of maturity are that it's difficult for business leaders to predict when new software services will be released to their clients, and the teams that construct these services will usually deliver them later than predicted or expected from the clients. Clients will also frequently encounter regressions in functionality or system errors caused by mistakes in manual processes. As a result, organizations with an initial level of DevOps maturity will find their capacity to innovate through custom software severely constrained (Humble and Molesky, 2011). At the managed level of DevOps maturity, software development teams are able to work with business leaders to set release time boxes. Software development teams then vary the scope of the work performed and the effort applied to meet the agreed release date. At the managed level, delivery time boxes are usually still outside the bounds of what the business may need. As a result, a managed level of DevOps maturity is characterized by frequent negotiations over the priority of requirements, defects, and system capabilities versus budget and the release window. Because this negotiation process continues throughout the release, it's critical to have an identified product owner who can examine functionality and quality trade-offs and maintain customer support for the service while it's being developed.

At this level, business sponsors regularly participate in scope/resource/date trade-offs and are well informed about the progress of service development. Accordingly, development teams should be capable of setting a release date for a service and then managing their development schedule to meet that date (more or less). Business sponsors may choose to slip a release date if the scope of the project changes or unforeseen risks occur that need to be dealt with before deploying the service. The development team's basic release capability at the managed level, gives business sponsors some visibility into service development and provides limited opportunities for course corrections and re-prioritization of requirements for subsequent development releases (Sacks, 2012). While the speed of service releases increases under a managed state of DevOps maturity, there is still enough latency to prevent release on demand. Projects that implement Agile development practices like regular integration and backlog management should find that they have the necessary process discipline to achieve a managed state, where even if they can't release new capability as fast as business sponsors need it, they can demonstrate regular process and confirm that an envisioned service will function as planned while meeting desired business objectives and performance measures (Samer, 2015). At the defined level of DevOps maturity, the release process becomes a regular key indicator of project health. Most development teams that operate at this level of maturity create a build at least once a day from trunk, and developers will ensure they don't have more than a day's work sitting on a local branch in version control. Accordingly, branches to the main release trunk are short-lived, and the result is a significant reduction in system integration issues which tend to fester when individual developers or sub-teams maintain private builds or source code branches. At the defined level of maturity, deployment of new service versions is further accelerated by automating the process of provisioning integrated environments. In most cases, development teams at this level focus on automating deployment into a system test environment. When a team reaches the defined state of maturity, the result is a regular release cadence. The release time box is well defined, and it's easy to identify early warning sign that a project is in trouble. This makes it possible to exert early corrective actions like scope reduction, resource augmentation, and a schedule re-plan. While service delivery is much more predictable than the initial or managed level of maturity, it may still not be as fast as business leader's desire.

At the measured level DevOps of maturity, they reach a critical plateau where they can deploy a new release whenever one is needed. From a business perspective, the speed of service development meets or exceeds that capacity of the business to assimilate new services. At this level of maturity, software development team no longer hinders business innovation, but on the other side it enables it. As development teams reach this level of maturity, a shift in organization structure. Instead of vertically organized centres of excellence (e.g., business analysts, development, quality assurance/testing, infrastructure and operations/support), we see an organizational shift to cross-functional product teams. Individual roles become less important than the tasks that need to be completed to release new capability/service, and teams are likelier to self-organize their works. As part of this transition, the entire team assumes responsibility for service-level agreements associated with their product. And it's not just service testing and deployment that folds into the cross-functional organization. User experience should also be integrated into development teams at this level. At that level of maturity, teams find that they have eliminated all remaining bottlenecks in downstream build, test, and deployment phases. Deployments of individual changes in source code can now be performed in minutes, and there's no reason a team can't do multiple deployments to production in a single day. As software development teams move to this level of maturity, delivery teams prioritize keeping the main code trunk deployable over doing new work. Development teams don't let the integration build stay broken for more than a few minutes and anybody is empowered to revert breaking changes from version control.

The focus of testing services also shifts up the development process. Test-driven development and acceptance-test-driven-development become core processes for development teams. These tests are layered on top of pre-flight builds to provide semantic validation of system function in addition to technical validation. At the optimizing level of DevOps maturity, software development teams drive a continuous stream of incremental innovation. They form hypotheses about customer needs and how they can serve them, run experiments to test these hypotheses with customers, and then use feedback from their experiments to make design and service implementation decisions based on the best course of action. In this hypothesis-driven development model, software development teams focus on optimizing cycle time in order to learn from customers in production, by gathering and measuring system feedback. Asynchronous services allow load balancing work distribution

among different variants of the same service. Software development teams are able to inject probes into real-time production operations to monitor application load, deploying more resources as necessary at this level. Developers also test and evaluate the system as it functions in production — they may even initiate failures to make sure that the system takes proper corrective action. At the optimizing level, database changes are decoupled from application deployments, and individual service endpoints are decoupled from each other (Komi-Sirviö and Tihinen, 2013). Software development teams at this level of DevOps maturity, can initiate business change by running experiments and proving business value. As the cost of each service deployment trends toward zero, the cost of running discrete experiments also drops significantly and becomes largely a factor of development labor costs.

Proposed transformation framework

The proposed framework based on the GQM (Goal Question Metric) approach (Sacks, 2012). First step, is to clarify the goal/s to be achieved from the transformation with respect to various models of quality and relative to particular environment. Second, set of questions is used to define models of the object of study and then focuses on that object to characterize the assessment or achievement of a specific goal. Third and last step, a set of metrics, based on the models, is associated with every question in order to answer it in a measurable way. These metrics/measurements associated with each goal should reflect/influence what matter to business and map this then to activities/tasks done by individual/s to lead to this influence. The transformation framework consists of around 14 capabilities that form the major components where any organization should adopt to follow DevOps delivery model. These capabilities vary between operational, delivery, governance, management, communication and process aspects. Each one of these capabilities assessed against different criteria to measure the different dimensions of the corresponding capability. DevOps maturity levels described in the previous section is utilized to measure the maturity level of each capability criteria. Each criteria has a standard description against each maturity levels (level 1 to level 5) where organization or project should meet to achieve such maturity level of the corresponding capability criteria. To summarize the list of transformation capabilities, a brief description for each one along with the corresponding criteria will be listed as follows:

1. Operational management

- Incident management methodology
- Responsible teams
- Troubleshooting and incident analysis approach
- Incident monitoring and communication tracking
- Incident notification and alerts mechanism
- Incident response strategy.

2. Service management

- Service management strategy
- Adopting standards sharing approach
- Collaboration cross teams
- Communication style cross teams
- Delivery model cross teams

3. Governance and process management

- Service delivery model is available and up-to-date (including engagement model, org chart, team structure, R&R)
- Service operation model is available and is reflecting the operation for the specific service
- The governance model for service delivery is defined and standardized (including RACI model, accountability)
- Standard SOW is available and is specific for the service
- Standard SLA is available and is specific for the service
- Estimation techniques are standardized

4. Build and continuous integration management

- Build strategy
- Build and integration cross environments
- Versioning control for the build artifacts
- Reporting mechanism/hierarchy
- Build automation
- Automation approach for continuous deployments/integration
- Provisioning strategy
- Deployable status to mainline

5. Tools and automation

- Tooling matrix for build and run is defined and standardized
- Tools packages are identified for large, medium and small implementations (including tiers of tools, implementation plan, price & cost)
- Service automation and tooling support business objectives
- Tool matrix for automation is defined, and its implementation plan is established

6. Quality assurance and testing management

- Testing strategy
- Testing Automation and responsibility
- Testing phase involvement within development life cycle
- Development involvement through testing
- Release cycle impact with testing
- Regression bugs exists

7. Project and delivery management

- Project management strategy
- Delivery approach
- Prioritization for operational versus new features
- Collaboration support cross teams handling approach
- R&R (Roles and Responsibilities) is clear for all stakeholders

8. Collaboration and communication management

- Organization/process model
- Communication style
- Collaboration style between Dev/Ops
- Level of continuous improvement
- Team organization
- Process documentation
- Reporting hierarchy
- Metrics and measurement techniques

9. Feedback and continuous improvement

- Feedback loops strategy
- Monitoring and alerting applied on which components
- Service monitoring metrics applied level
- Alerting and monitoring scope
- Service failures proactive versus reactive

10. Vendor management

- Vendor management strategy
- Services management approach
- Handling of incident management

Table 1. Project current state for Change management capability criteria

Criteria	Project behavior/state
Change management strategy	Cross functional
Change management control board formulation	Sign-off by CCB (Change Control Board) is mandatory
Change automation	Automation is applied for new artifacts via switches constantly
Review process for each change	Changes are not applied to main line unless being approved/tested on non-production environments by authorized persons
Feedback loops implemented	Application changes only automated where no feedback loops implemented
Data migration strategy	Manually handled

Table 2. Project level for Change management capability criteria

Criteria	Project behavior/state
Change management strategy	Level 4
Change management control board formulation	Level 1
Change automation	Level 5
Review process for each change	Level 5
Feedback loops implemented	Level 1
Data migration strategy	Level 1

Table 3. Actions, CSFs, KPIs for the Change management capability criteria

Criteria	CSFs	KPIs	Actions
Change management strategy	Not applicable	Not applicable	Not applicable
Change management control board formulation	Decentralized auditing approach for the change management process 1	%Reduction in the review time for any change	The successful adoption of DevOps methodologies often entails a close look at organizational structures, roles and responsibilities where teams have a broader view and responsibility for the entire release process rather than individual roles
Change automation	Time to market is very critical	#succeeded deployments versus total number of rollouts on all environments	As developers increasingly define infrastructure resources via code, operations require the capability to understand capacity requirements for the applications as they are being built. By integrating application performance management data with capacity planning tools, teams can reduce waste by avoiding overspending on unnecessary infrastructure, reduce risk and guarantee service delivery
Review process for each change	Decentralized auditing approach for the change management process	%Reduction in the review time for any change	Highly mature release and deployment practices incorporate comprehensive views of release trends, enabling managers to monitor and audit the entire deployment process. This should be supplemented with process interaction during execution for real-time remediation
Feedback loops implemented	How fast user feedbacks are incorporated in the development cycle	I increased customer and user satisfaction with plans and communications	Increasingly teams collaborate towards building a shared set of metrics, testing and release processes associated with meeting business outcomes. These may include transaction counts and web-site performance to meet conversion rates and reducing lead times to meet business demand for new services
Data migration strategy	Ensure data migrations are clearly integrated with the application deployments with clear rollback plans	Increased score in surveys of customer, user and service operation function satisfaction with release and deployment management	Include the data migrations into the deployment track for any application to minimize probability for roll backs

11. Continuous deployment management

- Deployment approach
- Fully automated
- Data based related deployment approach
- Release cycle and cadence
- Deployment cross environments mechanism

12. Configuration management

- Configuration management strategy
- Versioning control for the environments
- Configuration Items control and management

13. Technology and architecture management

- Technology style
- Custom versus rigid based
- Static versus dynamic

14. Change management

- Change management strategy
- Change management control board formulation
- Change automation
- Review process for each change
- Feedback loops implemented
- Data migration strategy.

Framework validation

To put the transformation framework into action, a use case is used to show how the transformation model can be utilized via practical case from within one of the organization running projects (Fitzpatrick and Dillon, 2011). I preferred to pick a sample critical capability like 'change management' and try to assess the project current change management process/behavior against the transformation framework standard capability/criteria. The assessment is done by matching the observed/current project/organization behavior against the capability/criteria levels behavior/description as detailed in the transformation framework. The closest match will indicate the current maturity level of this capability criteria. This process will then be repeated against the other criteria related to each and every capability that mentioned in the previous section. The outcomes from this initial assessment will be considered as gap assessment for the organization/project around the current DevOps maturity level. It can be then used to draft next milestones to move towards the next maturity level of the transformation towards the final goal which is reaching optimized level of maturity to maximize the outcomes from adopting the DevOps model. To show how the above mentioned process can be applied, I will assume the current project state for the change management capability criteria is as follows:

Applying the process described in the previous sections, the DevOps process auditor will first use the transformation framework to assess the current project/account state based on gap assessment against each capability criteria. I'll show this for the change management capability and the same process can be applied similarly against the rest of other 13 capabilities. Thus the process will follow the below mentioned workflow steps:

- Compare each and every capability criteria against the project behavior for the same criteria as given in Table 1.
- Find the best and close match between the project behavior for specific criteria and level of the corresponding criteria in the transformation framework.
- Assign the matched level to selected criteria.
- Apply the above process for all the capability criteria to get the maturity level and current state for each and every capability criteria.

Start with the change management capability criteria mentioned in Table 1, and use the above mentioned process, you will reach to the criteria levels as described in Table 2. The results shown in Table 2 clarify the current project status measured against the DevOps transformation framework. Where the project is very high mature (level 5) in some criteria like (Change automation, Review process), high mature in change management strategy (level 4) and still building up the skills/caliber for change management control board, feedback loops implementation and data migration strategy criteria. These results can be used by the project team to draft the future action plan to build on for those criteria/areas where at lower maturity levels. The framework helps the project team to build the action plan to cover the current gap by proposing set of actions along with set of assessment criteria or CSFs (Critical Success Factors) and KPIs (Key Performance Indicators) to measure the progress and clarify clear objective/goal towards building high mature criteria for all the capabilities. To show how this applies for the change management capability, a sample of set of actions, CSFs, KPIs, will be described in Table 3.

The table shows the data for each and every criteria within the change management capability. Project team can build on the given CSFs, KPIs, and proposed improvement actions for each and every capability to move towards higher DevOps maturity level and transform their current delivery practices by adopting the best practices and standards proposed by the DevOps transformation framework. To show for example as per Table 3, Change management automation criteria shows that time to market is the Critical Success Factor (CSF) along with number of succeeded deployments versus the total number of deployments within specific range of time is considered as Key Performance Indicator (KPI) to measure the performance/progress of the project against clear quantitative metric. This process can then be repeated regularly every six months or so based on the maturity of the organization and project budget to apply such transformations while satisfying the delivery commitments towards business users and clients.

Conclusion

This research work presents a new approach for both measuring/assessing the DevOps maturity level of any organization/project using set of capabilities/criteria and helping the organization/project to transform their current

maturity level based on the outcomes from the initial gap assessment. The uniqueness of the proposed framework is three folds, first the simplicity of the proposed mechanism to assess and transform based on the goals set by the organization/project in quantifiable or metric approach. Second, adopting quantitative or metric approach to measure the performance and progress through the transformation process for each and every capability. Third, Using CSF, KPIs, and set of improvement actions that guide the project/organization to move towards higher maturity level in smooth and seamless manner. Future plan based on work introduced in this paper will be focused to build on this transformation framework and design a tool to calculate the overall maturity level of each organization/project based on the assessment outcomes using simple GUI algorithm with user friendly interface that can be used by any process owner or organization aims to invest on their DevOps capabilities. The objective of this work is basically target to standardize the DevOps process/best practices to be adopted by most of current IT organization due to the main value/need expected from DevOps to fulfil current market demand (Loukides *et al.,* 2012).

REFERENCES

Akerele, O., M. Ramachandran, and M. Dixon. System dynamics modeling of agile continuous delivery process. In Proceedings - AGILE 2013, pages 60(63), 2013.

Bang, S., S. Chung, Y. Choh, and M. Dupuis. A grounded theory analysis of modern web applications: Knowledge, skills, and abilities for devops. In RIIT 2013 - Proceedings of the 2nd Annual Conference on Research in Information Technology, pages 61(62), 2013.

Bass, L., R. Je_ery, H. Wada, I. Weber, and L. Zhu. Eliciting operations requirements for applications. In 2013 1st International Workshop on Release Engineering, RELENG 2013 - Proceedings, pages 5(8), San Francisco, CA, 2013.

Cukier, D. Devops patterns to scale web applications using cloud services. In Proceedings - SPLASH '13, pages 143{152, Indianapolis, Indiana, USA, 2013.

Debois, P. Opening statement. *Cutter IT Journal*, 24(8):3{5, 2011.

DeGrandis, D. Devops: So you say you want a revolution? Cutter IT Journal, 24(8):34{39, 2011.

Feitelson, D., E. Frachtenberg, and K. Beck. Development and deployment at facebook. IEEE Internet Computing, 17(4):8{17, 2013.

Fitzpatrick, L. and M. Dillon. The business case for devops: A five-year retrospective. *Cutter IT Journal*, 24(8):19(27), 2011.

Heiko Koziolek. Goa, Question Metric, solume 4909 of the series lecture notes in computer science pp39-42.

Hosono, S. and Y. Shimomura. Application lifecycle kit for mass customization on PaaS platforms. In Proceedings - 2012 IEEE 8th World Congress on Services, SERVICES 2012, pages 397(398), Honolulu, HI, 2012.

Humble, J. and J. Molesky. Why enterprises must adopt devops to enable continuous delivery. *Cutter IT Journal*, 24(8):6(12), 2011.

Jez Humble, Chris Read, Dan North, The Deployment Production Line, Proceedings of Agile 2006, IEEE Computer Society http://h30499.www3.hp.com/t5/ Business-Service-Management-BAC/DevOps-and-Ops Dev-How-Maturity-Model-Works/ba-p/6042901#.VNgkE KP8IiQ (Available Jan 2015)

Keyworth, B. Where is it operations within devops? Cutter IT Journal, 24(12):12(17), 2011.

Kitchenham, B. Procedures for performing systematic reviews, 2004.

Komi-Sirviö, S, and Tihinen, M. 2003. Great Challenges and Opportunities of Distributed Software Development - An Industrial Survey. In proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering, SEKE2003, San Francisco, USA pp. 489 – 496

Le-Quoc, A. Metrics-driven devops. Cutter IT Journal, 24(12):24(29), 2011.

Loukides, M. What is DevOps? O'Reilly Media, Sebastopol, CA, 2012.

Neely, S. and S. Stolt. Continuous delivery? easy! Just change everything (well, maybe it is not that easy). In Proceedings - AGILE 2013, pages 121(128), 2013.

Phifer, B. Next-generation process integration: CMMI and ITIL do devops. Cutter IT Journal, 24(8):28{33, 2011.

Pruijt, H. Multiple personalities: the case of business process reengineering. *Journal of Organizational Change Management*, 11(3):260(268), Jan. 1998.

Roche, J. Adopting devops practices in quality assurance. Communications of the ACM, 56(11):38(43), 2013.

Sacks, M. DevOps principles for successful web sites. In Pro Website Development and Operations. Springer, 2012.

Samer, I. Mohamed DevOps shifting software engineering strategy - value based perspective, manuscript Volume 17, Issue 2, Ver. IV (Mar – Apr. 2015), PP 51-57.

Schaefer, A., M. Reichenbach, and D. Fey. Continuous integration and automation for devops. Lecture Notes in Electrical Engineering, 170 LNEE:345(358), 2013.

Shang, W. Bridging the divide between software developers and operators using logs. In Proceedings - International Conference on Software Engineering, pages 1583(1586), 2012.

Stuckenberg, S., E. Fielt, and T. Loser. The impact of software-as-a-service on business models of leading software vendors: Experiences from three exploratory case studies. In PACIS 2011 - 15th Paci_c Asia Conference on Information Systems: Quality Research in Pacic, 2011.

Tessem, B. and J. Iden. Cooperation between developers and operations in software engineering projects. In Proceedings - International Conference on Software Engineering, pages 105(108), 2008.

Walls, M., Building a DevOps Culture. O'Reilly Media, Sebastopol, CA, 2013.

Webster, J. and R. T. Watson. Analyzing the past to prepare for the future: Writing a literature review. MIS Q., 26(2): xiii xxiii, June 2002.

****